(19) Japan Patent Office (JP)	(12) Japanese Unexamined Patent Publication (A)	(11) Japanese Unexamined Patent Publication No. H7-225770
		(43) Public Patent Disclosure Bulletin Date: August 22, 1995

(51) Int. Cl.^{6}	Identification code	Internal File No.	FI	Tech. Indic.
G 06 F 17/30		9194-5L	G 06 F 15/419 310	

Request for Examination: N ot yet made
Number of claims: 3 FD
Total pages: 15

(21) Patent Application No.: H6-36346
(22) Filing Date: February 10, 1994

(71) Applicant::000005496
Fuji Xerox Co., Ltd.
3-3-5 Akasaka, Minato-ku, Tokyo
(72) Inventor: Kenichi Numata
c/o Yokohama Business Park East Tower,
Fuji Xerox Co., Ltd., 134 Kobe-cho, Hodogaya-ku,
Yokohama City, Kanagawa Prefecture
(72) Inventor: Hideyuki Fujiwara
(74) Agent:: Sadao Minamino (and 3 others)

(54) [Title of the Invention] A Data Retrieval Apparatus

(57) [Abstract]
[Purpose] To provide a data retrieval apparatus to retrieve a node from a node set, in which the relation between nodes is expressed as a tree or a tree set, that satisfies a second condition and has a progenitor-descendant relation with a node satisfying a first condition.
[Constitution] An input means inputs a first condition, a second condition and a progenitor-descendant relation between them. A node retrieval means receives the first and second conditions and extracts a record set corresponding to a node from the index of an index retaining means that satisfies the first condition and a record set corresponding to a node that satisfies the second condition. A connection relation retrieval means receives the progenitor-descendant relation, checks whether or not a node pointed by one record exists in the subtree index pointed by the other record extracted by the node retrieval means, and, if the node exists and the progenitor-descendant relation is satisfied, outputs the node as a node that satisfies the second condition.

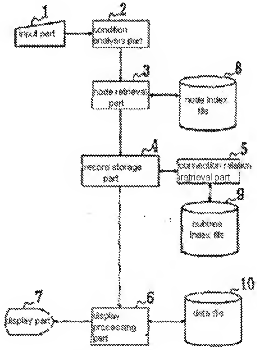


Fig 1. Constitution of a data retrieval apparatus

[Claims]

[Claim 1] A data retrieval apparatus that is used for retrieving a node from a node set, in which the relation between nodes is expressed as a tree or a tree set, that satisfies a second condition and has a progenitor-descendant relation with a node satisfying a first condition, said data retrieval apparatus comprising an input means to input the first condition, the second condition and a progenitor-descendant relation between them, a data storage means to store nodes, the relation among the nodes and attribute values of the nodes, a subtree index storage means to retain a subtree index provided for all the nodes stored in the data storage means except for leaf nodes that comprises subtrees in which said nodes are the root nodes thereof and pointers for nodes, a record set retaining means to retain a record set that are provided for nodes stored in the data storage means and comprises pointers for the nodes and pointers for the subtree index, an index retaining means to retain an index comprising attribute values of nodes stored in the data storage means and pointers for the records corresponding to nodes having the attribute values, a node retrieval means that receives the first and second conditions inputted from the input means and extracts from an index in the index retaining means a record set corresponding to nodes that satisfy the first condition and a record set corresponding to nodes that satisfy the second condition, a connection relation retrieval means that receives the progenitor-descendant relation between the first and second conditions, checks whether or not a node pointed by one record exists in the subtree index pointed by the other record extracted by the node retrieval means, and, if the node exists and the progenitor-descendant relation is satisfied, outputs the node as a node that satisfies the second condition and a display processing means that displays information of the node outputted from the connection relation retrieval means by extracting it from the data storage means.

[Claim 2] The data retrieval apparatus according to claim 1, wherein the subtree index storage means further comprises data for distinguishing nodes pointed by pointers in the subtree index as to whether each node is a child of a root node in the subtree index or a descendant other than a child.

[Claim 3] A data retrieval apparatus that is used for retrieving a node from a node set, in which the relation between nodes is expressed as a tree or a tree set, that satisfies a second condition and has a progenitor-descendant relation with a node satisfying a first condition, said data retrieval apparatus comprising an input means to input the first condition, the second condition and a progenitor-descendant relation between them, a data storage means to store nodes, the relation among the nodes and attribute values of the nodes, a record set retaining means to retain a record set provided for nodes stored in the data storage means that comprises node pointers for said nodes and parent pointers pointing at records that retain node pointers for parent pointers of said nodes, an

index retaining means to retain an index comprising attribute values of nodes stored in the data storage means and pointers for the records corresponding to nodes having the attribute values, a node retrieval means that receives the first and second conditions inputted from the input means and extracts from an index in the index retaining means a record set corresponding to nodes that satisfy the first condition and a record set corresponding to nodes that satisfy the second condition, a connection relation retrieval means that receives the progenitor-descendant relation between the first and second conditions inputted from the input means and outputs a node pointed by a node pointer found by calculating a product set between a set of node pointers for records obtained by tracing parent pointers for descendant records extracted by said node retrieval means back to root nodes and a set of node pointers for records that satisfy the second condition, and a display processing means that displays information of the node outputted from the connection relation retrieval means by extracting it from the data storage means.

[Detailed Explanation of the Invention]

[0001]

[Industrial Field of the Invention] The present invention relates to a data retrieval apparatus. Especially, the invention relates to a data retrieval apparatus that speeds up retrieval processing at a time when a node is retrieved from a node set in which the relationship between nodes are expressed as a tree or a tree set using node conditions and connection relation conditions between nodes.

[0002]

[Prior Art] Data processors have been developed that structuralize data using a tree structure in order to facilitate data processing according to its content. For example, document data handled by a document processor can be structuralized by a tree structure because the content of a document can be conceptually expressed by a logic structure having layers such as chapters, clauses and items.

[0003] The logic structure of documents is one example of data structures expressed by tree structures. The logic structure of documents can usually be expressed by a hierarchically directed tree as shown in Fig. 8. Fig. 8 shows one example of tree structures used for the logic structure of documents. In the example of the logic structure of a document as shown in Fig. 8, a document entitled by "report" has a logic structure comprising a root node (i.e., "report") and the first layer node having the title, names of authors, content of chapter 1, content of chapter 2, and content of chapter 3. The content of chapter 1 has a second layer node consisting of "a header" and three "paragraphs." The content of chapter 2 has a second layer node comprising "a header," a drawing and two "paragraphs." Likewise, the content of chapter 3 has a second layer node consisting of "a header," "a drawing" and two "paragraphs." This is the logic structure of the document.

[0004] In retrieving data from the logic structure of a document expressed by a tree structure, the data retrieval processing is performed by using not only the conditions of individual nodes, elements of tree-structure data, but also the conditions of a parent-child or progenitor-descendant relation. For example, in the logic structure of a document entitled "report," retrieval is performed under the following condition: "find a chapter including a character string "database" in the header directly below as well as a drawing.

[0005] In this case, "a character string 'database' in the header," "a chapter" and "a drawing" are conditions on nodes. The "directly below" and "including" are conditions determining relation between nodes such as a parent-child relation and a progenitor-descendant relation. Accordingly, there are multiple nodes that satisfy conditions on nodes such as "a character string 'database' in the header," "a chapter" and "a drawing" in this example, while there is only one node that satisfies conditions on the relation between nodes such as a parent-child relation and a progenitor-descendant relation, i.e., the node chapter 2. Thus, data retrieval can be narrowed down by retrieving not only conditions on nodes but also conditions on the relation between nodes. In the example as shown in Fig. 8, only the "chapter 2" is retrieved as a node that satisfies conditions.

[0006] In the aforementioned retrieval, an answer can be retrieved by finding a product set between a first set, i.e., "a chapter that has a header including a character string "database" directly below and a second set, i.e., "a chapter including a drawing in the lower order." In this case, the first set and the second set are retrieved by using conditions on individual nodes and conditions on a parent-child relation or a progenitor-descendent relation, respectively.

[0007] In the data structure expressed by a tree structure in which retrieval processing is performed using not only conditions on individual nodes but also conditions on a parent-child or progenitor-descendent relation between nodes, a node is retrieved that satisfies a second condition and has a parent-child or progenitor-descendent relation with nodes that satisfy a first condition.

[0008] Thus, the following retrieval processing method is generally used at a time when a high speed retrieval processing is performed in order to retrieve a node that satisfies a second condition and has a parent-child or progenitor-descendent relation with nodes that satisfy a first condition.

[0009] In the case of using a tree structure as a data structure, a data format is usually provided such that each node of data elements is expressed by record-type data and the state of linking between data expressed by a pointer. Thus, in the aforementioned data retrieval processing method, the simplest processing procedure is such that pointers showing the linking state of records are traced one by one in accordance with the

order (e.g., preferential order and width priority) determined by the data structure having a tree structure to examine if each record (node) satisfies a first condition or not and, if the first condition is satisfied, it is then examined if a second condition is satisfied or not by starting with the record and tracing pointers one by one to find a record that has a parent-child or progenitor-descendant relation.

[0010] In the case of speeding up the aforementioned retrieval processing, the processing is divided into a first processing in which a node is retrieved that satisfies the first condition and a second processing, starting with the node that satisfied the first condition, in which nodes are retrieved that has a parent-child or progenitor-descendant relation and satisfies the second condition, and then each retrieval processing is separately sped up.

[0011] Accordingly, in such a data retrieval processing method, a transposed file is provided in the first processing, for example, that allows for backward referencing, whereby a corresponding node is retrieved from all the nodes in a tee structure based on the attribute values of the nodes. Thus, a high speed retrieval processing is possible for a node that satisfies the first condition.

[0012] In order to speed up the second processing, a retrieval method can be used that is disclosed, for example, in "Chris Clifton and Hector Garcia-Molina, 'Indexing in a Hypertext Database,' Proceedings of 16th International Conference on VLDB, pp. 36-49, 1990." This is a method of providing transposed files for subtrees for all the nodes of a tree structure except for leaf nodes. Here, those nodes are the roots of those subtrees.

[0013] Fig. 9 is a diagram to explain a retrieval method using transposed files for subtrees at a time when retrieval processing is performed using a parent-child or progenitor-descendant relation of nodes. As shown in Fig. 9, in this retrieval method, transposed files 90a, 91a and 92a are provided for the root node 90, A node 91 and B node 92, respectively, as node data of the tree structure. Here, in the transposed files 90a, 91a and 92a, data is retained for the attribute values of nodes, pointers for the nodes having the attribute values, and connection relations (i.e., a parent-child relation or a progenitor-descendant relation) between those nodes and nodes having transposed files (roots of subtrees) so that backward reference is possible based on the attribute values of nodes.

[0014] For example, the transposed file 91a provided for the A node 91 retains data on the attribute values of nodes (bird, dog, fish) of a subtree whose root is the A node 91, pointer information for nodes having the attribute values (e.g., C node, F node, and E node) and the connection relation information (i.e., child or grandchild relation). The information on the connection relation is use to distinguish between a parent-child relation (c) and a progenitor-descendant relation (d).

[0015] The method of performing retrieval processing using transposed files for subtees as shown in Fig. 9 allows accelerating the second processing. That is, the method of starting with a node that satisfies a first condition and retrieving a node that satisfies a second condition having a child or grandchild relation therewith allows retrieving a node that satisfies the second condition with ease and at a high speed by retrieving the attribute nodes retained in transposed files provided for the starting node. This method allows for processing by far faster than a method of retrieving all the child or grandchild nodes one by one.

[0016] On the other hand, in the case of starting with a node that satisfies a first condition and performing the retrieval processing of a node that satisfies a second condition having a parent or progenitor relation therewith (e.g., "a node that satisfies a second condition and has a node in a child or a grandchild that satisfies a first condition"), retrieval processing is possible by using the method as shown in Fig. 9 if retrieval is performed for a node that satisfies a second condition first.

[0017] Besides the method of providing a transposed file for each subtree as shown in Fig. 9, methods of retrieving a node by starting with a node that satisfies a first condition and then retrieving a node that satisfies a second condition having a parent or progenitor relation therewith include a method of allowing each node to have a pointer for its parent node. In such a case, a data structure using a tree structure is bilaterally linked to each other.

[0018] Fig. 10 is a diagram showing an example of data structures having a path necessary for tracing back to a progenitor node from a certain node. As shown in Fig. 10, this data structure having a tree structure is bilaterally linked so that it is possible to start with a certain node and trace back to its parent or progenitor node. This method can further be divided into two modes depending on how node information is allocated to a secondary storage. First, a description of two modes is given below that differ in the way of allocating node information to a secondary storage.

[0019] In the first mode, as shown in Fig. 11, information on one node is allocated to a continuous region on a secondary storage. In the second mode, as shown in Fig. 12, information on one node is allocated to a secondary storage after it is divided into multiple regions.

[0020] In the data structure according to the first mode, information on one node is allocated to a continuous region on a secondary storage. That is, as shown in Fig. 11, information of one node 110 consists of a pointer 111 for a parent node, multiple pointers 112 for child nodes and m attributes 113. They are allocated to a continuous region on a secondary storage. In the attribute field are stored actual values retained in each node.

[0021] In the data structure according to the second mode, information on one node is allocated to a secondary storage after it is divided into multiple regions. In this case, there are

many ways to divide a group of information on one node. For example as shown in Fig. 12, a high-speed access processing can be performed by separating link information between nodes from information on the contents of nodes, since the tracing of links in a tree structure can be operated by reading only link information from a secondary storage. That is, information of one node is constituted only by link information between nodes. The link information 121 has a pointer 121a for a parent node, a pointer 121b for a child node set and a pointer 121c for an attribute set. Here, the child node set 122 pointed by the pointer 121b and the attribute set 123 pointed by the pointer 121c are separated from the link information between nodes and allocated to a region on a separate secondary storage.

[0022] A description of the way of retrieving a parent or progenitor node, which satisfies a given condition, from a certain node is given below with regard to the data structures of the aforementioned two modes.

[0023] In the first mode (Fig. 11), information 110 on a certain node is read into the main storage from a secondary storage. A pointer 111 for a parent node, which is one field of the node, is examined. Then, information on the parent node is read into the main storage from the secondary storage. Information on attributes 113 of the node that was read is examined to find if the given condition has been satisfied or not. In the case of retrieving a progenitor node, the aforementioned process must be repeated.

[0024] In the second mode (Fig. 12), information 120 (sic) on a certain node is read into the main storage from a secondary storage. In this case, only link information 121 between nodes is read. Then, a pointer 121a for a parent node, which is one field of the link information 121 between nodes, is examined, and the information on the parent node is read into the main storage from the secondary storage. Next, a pointer 121c for an attribute set, which is one field of the link information between nodes of the parent node that was read, is examined, and the corresponding attribute set is read into the main storage from the secondary storage. Finally, information on the attributes of the attribute set that was read is examined to find if the given condition has been satisfied or not. In the case of retrieving a progenitor node, the aforementioned process must be repeated.

[0025]

[Problems that the Invention is to Solve] In the method of providing transposed files for subtrees for all the nodes except for leaf nodes (here, those nodes are roots of those subtrees), as shown in Fig. 9, nodes are duplicated among transposed files, resulting in problems of an increase in the total size of the transposed files as well as an increase in the amount of memory required. Given that the number of a balanced tree is n, the number of records in transposed files explosively increases (n log n).

[0026] In the method of making each node have a pointer for its own parent node and access the parent node or a progenitor node using the pointer in order to examine if the condition has been satisfied or not, multiple nodes must be accessed one by one for retrieval, which entails unnecessary accesses before a node suitable for the condition is found, requiring an increased retrieval time. Also, as shown in Fig. 11, in the case that one node is allocated to a secondary storage in the first mode, reading the node into the main storage from a secondary storage requires an extended period of time especially when the size of information of the node is large, resulting in a further increase in retrieval time.

[0027] On the other hand, even in the second mode method of separating link information between nodes and allocating it to a separate secondary node as shown in Fig. 12, it is necessary to go through two processes for examining if the parent node of a certain node has satisfied a given condition: a process of obtaining a parent node by tracing a pointer for the parent node based on the certain node; and a process of obtaining information on the content of the node by tracing a pointer for a attribute set based on the parent node thus obtained. Accordingly, reading of data into the main storage from a secondary storage must be performed twice, with the result that retrieval time hardly decreases as a whole.

[0028] Therefore, the object of the present invention is to provide a data retrieval apparatus that allows retrieving at a high speed a second node that satisfies a second condition and has a progenitor-descendant relation with a node that satisfies a first condition from a node set in which the relation between nodes is expressed by a tree or a tree set.

[0029] The second object of the present invention is to provide a data retrieval apparatus that substitutes a subtree index exclusive of information on the attribute values of nodes for a transposed file that allows starting with a certain node that satisfies a first condition and retrieving a node that has a parent-child or progenitor-descendant relation with the certain node and satisfies a second condition, thus decreasing a memory capacity for indexing and enhancing a processing speed.

[0030] The third object of the present invention is to provide a data retrieval apparatus that decreases the number of steps of reading data into the main storage from a secondary storage in the case of starting with a certain node that satisfies a first condition and retrieving a node that is its parent or progenitor and satisfies a second condition, thus allowing accessing a parent or progenitor node at a high speed.

[0031]

[Means of Solving the Problems] In order to achieve the aforementioned objects, the data retrieval apparatus according to the present invention (claim 1) is to provide a data retrieval apparatus that is used for retrieving a node from a node set, in which the relation between nodes is expressed as a tree or a tree set, that satisfies a second condition and has a

progenitor-descendant relation with a node satisfying a first condition, and that comprises an input means (1, 2) to input the first condition, the second condition and a progenitor-descendant relation between them, a data storage means (10) to store nodes, the relation among the nodes and attribute values of the nodes, a subtree index storage means (9) to retain a subtree index provided for all the nodes stored in the data storage means except for leaf nodes that comprises subtrees in which the above-said nodes are the root nodes thereof and pointers for nodes, a record set retaining means (14) to retain a record set that are provided for nodes stored in the data storage means and comprises pointers for the nodes and pointers for the subtree index, an index retaining means (12) to retain an index comprising the attribute values of nodes stored in the data storage means and pointers for the records corresponding to nodes having the attribute values, a node retrieval means (3, 4) that receives the first and second conditions inputted from the input means and extracts from an index in the index retaining means a record set corresponding to nodes that satisfy the first condition and a record set corresponding to nodes that satisfy the second condition, a connection relation retrieval means (4, 5) that receives the progenitor-descendant relation between the first and second conditions, checks whether or not a node pointed by one record exists in the subtree index pointed by the other record extracted by the node retrieval means, and, if the node exists and the progenitor-descendant relation is satisfied, outputs the node as a node that satisfies the second condition, and a display processing means (6, 7) that displays information of the node outputted from the connection relation retrieval means by extracting it from the data storage means.

[0032] Furthermore, in addition to the aforementioned constitution, the data retrieval apparatus according to the present invention (claim 2) is characterized in that the subtree index storage means further comprises data for distinguishing nodes pointed by pointers in the subtree index as to whether each node is a child of a root node in the subtree index or a descendant other than a child.

[0033] Also, the data retrieval apparatus according to the present invention (claim 3) is to provide a data retrieval apparatus that is used for retrieving a node from a node set, in which the relation between nodes is expressed as a tree or a tree set, that satisfies a second condition and has a progenitor-descendant relation with a node satisfying a first condition, and that comprises an input means (1, 2) to input the first condition, the second condition and a progenitor-descendant relation between them, a data storage means (10) to store nodes, the relation among the nodes and attribute values of the nodes, a record set retaining means (14) to retain a record set provided for nodes stored in the data storage means that comprises node pointers for said nodes and parent pointers pointing at records that retain node pointers for parent pointers of said nodes, an index retaining means to retain an index comprising attribute values of nodes stored in the data storage means and pointers for the records corresponding to nodes having the attribute values, a node retrieval means (3, 4) that receives the first and second conditions inputted from the input means and extracts from an index in the index retaining means a record set corresponding to nodes that satisfy the first condition and a record set corresponding to nodes that satisfy the second condition, a connection relation retrieval means (4, 5) that receives the progenitor-descendant relation between the first and second conditions inputted from the input means and outputs a node pointed by a node pointer found by calculating a product set between a set of node pointers for records obtained by tracing parent pointers for descendant records extracted by said node retrieval means back to root nodes and a set of node pointers for records that satisfy the second condition, and a display processing means (6, 7) that displays information of the node outputted from the connection relation retrieval means by extracting it from the data storage means.

[0034]

[Operation of the Invention] In the data retrieval apparatus according to the present invention (claim 1), at a time of retrieving a second node from a node set, in which the relation between nodes is expressed as a tree or a tree set, that satisfies a second condition and has a progenitor-descendant relation with a node satisfying a first condition, the input means (1, 2) first inputs the first condition, the second condition and a progenitor-descendant relation between them. The data storage means (10) stores nodes, the relation between nodes, and the attribute values of nodes. The subtree index storage means (9) is provided for nodes except for leaf nodes stored in the data storage means (10) and retains a subtree index comprising pointers for nodes constituting subtrees in which those nodes are root nodes thereof.

[0035] The record set retaining means (14) retains a record set for nodes stored in the data storage means (10). Each record comprises a pointer for the corresponding node and a pointer for the subtree index. The index retaining means (12) retains an index comprising the attribute values of nodes stored in the data storage means and pointers for the records corresponding to nodes having the attribute values.

[0036] After the node retrieval means (3, 4) have received the first and second conditions inputted from the input means and have extracted a record set corresponding to nodes that satisfied the first condition and a record set corresponding to nodes that satisfied the second condition from the index in the index retaining means, the connection relation retrieval means (5, 4) receives a progenitor-descendant relation between the first and second conditions inputted from the input means and examines if a node pointed by one record exists in the subtree index pointed by the other record extracted by the node retrieval means or not. If the corresponding node exists and the aforementioned progenitor-descendant relation is satisfied, the node will be outputted as a node that has satisfied the second condition. Depending on the result of the output, the display processing means (6, 7) extracts information of the node outputted from the connection relation retrieval means from the data storage means and displays it. Here, it is not necessary to retain the attribute values of nodes in the subtree index, resulting in a decrease in a memory capacity.

[0037] Also, in the data retrieval apparatus according to the present invention (claim 2), data is included that can distinguish nodes pointed by pointers in the subtree index retained in the subtree index storage means as to whether each node is a child of a root node in the subtree index or a descendant other than a child. Thus, it is possible to distinguish a child from a descendant other than a child at the time of the retrieval of a node.

[0038] In the data retrieval apparatus according to the present invention (claim 3), at the time of retrieving a second node from a node set, in which the relation between nodes is expressed as a tree or a tree set, that satisfies a second condition and has a progenitor-descendant relation with a node satisfying a first condition, the input means (1, 2) inputs the first condition, the second condition and a progenitor-descendant relation between them. The data storage means (10) stores nodes, the relation between nodes, and the attribute values of nodes. The record set retaining means (14) retains a record set for nodes stored in the data storage means (10). Here, the record comprises a node pointer for the corresponding node as well as a parent pointer pointing at a record that retains a node pointer for the parent node of the above-said node.

[0039] The index retaining means (12) retains an index comprising the attribute values of nodes stored in the data storage means and pointers for the records corresponding to nodes having the attribute values. Therefore, the node retrieval means (3, 4) receives the first and second conditions inputted from the input means and extracts a record set corresponding to nodes that satisfies the first condition and a record set corresponding to nodes that satisfies the second condition from the index in the index retaining means. Also, the connection relation retrieval means (5, 4) receives a progenitor-descendant relation between the first and second conditions inputted from the input means and outputs a node pointed by a node pointer found by calculating a product set between a set of node pointers for records obtained by tracing parent pointers for descendant records extracted by said node retrieval means back to root nodes and a set of node pointers for records that satisfy the second condition. Depending on the result of the output, the display processing means (6, 7) extracts information of the node outputted from the connection relation retrieval means from the data storage means and displays it.

[0040] Thus, in the data retrieval apparatus according to the present invention (claim 1), a subtree index is provided in order to check whether or not either a node that satisfies a first condition or a node that satisfies a second condition exists in a subtree in which one node is the root node of the other. In the conventional transposed file is included data on the attribute values of nodes, as shown in Fig. 9. On the other hand, the present subtree index need not include data on the attribute values of nodes in the index, resulting in a decrease in the memory capacity of the index.

[0041] Also, in the data retrieval apparatus according to the present invention (claim 3), a record comprising a set of a pointer for a node and a pointer for a subtree index in which the above-said node is the root node thereof has a pointer for a record corresponding to the parent node of the node corresponding to the record comprising the set. Thus, the retrieval of the parent node or a progenitor node of a certain node can be performed only by scanning a record set, resulting in a decrease in retrieval time for retrieving a progenitor node. Since the size of a record is set in advance and small enough so that time required for reading can be shortened as compared with conventional data management methods as shown in Fig. 11 and Fig. 12.

[0042]

[Embodiments] A description of one embodiment of the present invention is given below through reference to drawings. Fig. 1 is a block diagram showing the constitution of the main part of the data retrieval apparatus according to one embodiment of the present invention. In Fig. 1, the reference numeral 1 refers to an input part such as a keyboard, 2 a condition analysis part, 3 a node retrieval part, 4 a record storage part, 5 a connection relation retrieval part, 6 a display processing part, 7 a display part such as a display device, 8 a node index file, 9 a subtree index file and 10 a data file.

[0043] The node index file 8 retains an index used for retrieving a node that satisfies a given condition. The subtree index file 9 retains an index for a subtree in which each node is its root. The data file 10 retains a data set expressed by a tree structure.

[0044] After the input part 1 designates conditions for retrieval (i.e., a first condition, a second condition and a parent-child or progenitor-descendant relation condition), the condition analysis part 2 analyzes the condition designated by the input part 1. The node retrieval part 3 reads the node index file 8 based on the conditions analyzed by the condition analysis part 2 in order to retrieve nodes that satisfy the conditions. The record storage part 4 stores a record set retrieved by the node retrieval part 3 and the connection relation retrieval part 5. The connection relation retrieval part 5 reads a subtree index from the subtree index file 9 to retrieve a parent-child or progenitor-descendant relation from the record set retained in the record storage part 4. The display processing part 6 reads the corresponding node from the data file 10 based on the result of the retrieval retained in the record storage part 4 and performs a display processing for the display part 7. As a result,

the result of the retrieval is displayed on the display part 7.

[0045] Next, the data structures are explained in detail below for the node index file 8, the subtree index file 9 and the data file 10. Fig. 2 is a diagram showing the data structures of the node index file, the subtree index file and the data file as well as the relationship among them. As shown in Fig. 2, the node index file 8 consists of an index 12 and a record set 14. The index 12 retrieves records from the record set that have pointers for nodes that satisfy given conditions. The index 12 allows retrieving one node by designating multiple conditions.

[0046] An individual record 13 constituting the record set 14 consists of a pointer 15 for a parent record, a pointer 16 for a subtree index and a pointer 17 for a node, thereby relating the parent record, the subtree index and the node. In the case that a node pointed by a pointer of a record 13 has no parent node, i.e., the node is a root node, a pointer 15 for the parent node is vacant. Similarly, in the case that a node pointed by a pointer of a record 13 is a leaf node, a pointer 16 for a subtree index is also vacant.

[0047] The subtree index file 9 retains a subtree index 11 for a subtree whose root is each node. The subtree index 11 is an index for a subtree whose root is a node pointed by a pointer for the node in a record 13 and comprises a set of a connection relation 18 and a pointer 19 for a node. The subtree index 11 allows retrieving a connection relation 18 based on a set of a connection relation 18 and a pointer 19 for a node by using a pointer for a node in a certain record. Data on connection relation 18 retains a connection relation between a node and the corresponding record by distinguishing a child from a progenitor other than a child using symbol codes "C" and "D," respectively.

[0048] The data file 10 retains a set of node data 21, which is a data set expressed by a tree structure. The node data 21 comprises pointers showing the link state of nodes in a tree structure and their attributes, i.e., data of the main body part (See Fig. 11 and Fig. 12).

[0049] Next, a description of retrieving a node is given below that satisfies a second condition and is a child or a descendant of a node that satisfies a first condition using record data of the node index file 8, the subtree index file 9 and the data file 10 having such a data structure. Fig. 3 is a flow chart showing an example of data retrieval processing for retrieving a node that satisfies a second condition and is a child or a descendant of a node that satisfies a first condition.

[0050] A description of data retrieval processing is given below through reference to Figs. 1, 2 and 3. After starting the processing, the input part 1 reads a first condition, a second condition and a connection relation condition for nodes in a step 201. In a step 202, the condition analysis part 2 analyzes the first condition, the second condition and the connection relation condition for nodes read by the input part. In a step 203, the node retrieval part 3 reads an index 12 from the node index file 8. In a step 204, the node retrieval part 3 retrieves nodes that satisfy the first condition based on the conditions analyzed by the condition analysis part 2 using the index 12. As a result, a record set thus obtained is retained in the record storage part 4.

[0051] Next, in a step 205, the node retrieval part 3 similarly retrieves nodes that satisfy the second condition based on the conditions analyzed by the condition analysis pat 2 using the index 12 and retains a record set thus obtained in the record storage part 4. In a step 206, a pointer for a node on the descendant side is checked within a subtree index for a node on the progenitor side. That is, a pointer for a node in a record of a record set that has satisfied the second condition and has been retained in the record storage part 4 in the previous step 205 is retrieved from within a subtree index pointed by a record of a record set that has satisfied the first condition and has been retained in the record storage part 4 in the previous step 204.

[0052] In a step 207, it is checked if the corresponding node is found or not. If the node is not found, the procedure is advanced to a step 211 to check if the retrieval has been completed for all the records or not. If any unprocessed node remains, the procedure is advanced for the next node. If the corresponding node is found in the above step, the procedure is advanced to a step 208 to further check if the connection relation is satisfied in the corresponding node. If the connection relation is not satisfied in that step, the procedure is advanced to a step 211, as in the previous step 207, to check if the retrieval has been completed for all the records or not. If any unprocessed node remains, the procedure is advanced for the next node. That is, it is checked if a pointer for a node is found in a subtree index or not and a designated connection relation is satisfied or not.

[0053] After it is confirmed that a pointer for a node is found in a subtree index and the designated connection relation has been satisfied, the procedure is advanced to a step 209 to read information on the corresponding node from the data file, and, in the following step 210, information on the content of the node is displayed on the display part. Next, the procedure is advanced to the step 211 to check if the retrieval has been completed for all the records or not. If any unprocessed node remains, the procedure is returned to the step 206 to go through the steps starting with the step 206 for the next node. A series of processes is finished if it is determined that the retrieval is completed for all the records in the step 211.

[0054] Thus, in this data retrieval processing method, it is checked with a subtree index as to whether or not either a node that satisfies a first condition or a node that satisfies a second condition exists in the subtree index in which one node is the root of the other. Thus, a node is retrieved that satisfies first and second conditions. This method eliminates the necessity of including data on the attribute values of nodes in an index, which allows for data retrieval with a decrease in a

memory capacity for indexing.

[0055] In the data retrieval apparatus according to the present invention, as shown in Fig. 2, an individual record 13 of the record set 14 has not only a set of a pointer 17 for a node and a pointer 16 for the subtree index in which the node is the root thereof, but also a pointer 15 for a record corresponding to the parent node of the node corresponding to the above-said individual record 13. Thus, the parent node or a progenitor node of a certain node can be retrieved only by scanning a record set.

[0056] A description of the data retrieval processing is given below in the case of using a pointer 15 for a parent record of an individual record 13. Fig. 4 is a flowchart showing a procedure of the data retrieval processing for a node that satisfies a second condition and is a parent or a progenitor of a node that satisfies a first condition using a pointer 15 for a parent record of an individual record 13.

[0057] As in the previous case, a description of the data retrieval processing is given below through reference to Figs. 1, 2 and 4 in the case of retrieving data only by scanning a record set. After starting the processing, the input part 1 reads a first condition, a second condition and a connection relation condition for nodes in a step 301. In a step 302, the condition analysis part 2 analyzes the first condition, the second condition and the connection relation condition for nodes read by the input part. In a step 303, the node retrieval part 3 reads an index 12 from the node index file 8. In a step 304, the node retrieval part 3 retrieves nodes that satisfy the first condition based on the conditions analyzed by the condition analysis part 2 using the index 12. As a result, a record set thus obtained is retained in the record storage part 4.

[0058] Next, in a step 305, the node retrieval part 3 similarly retrieves nodes that satisfy the second condition based on the conditions analyzed by the condition analysis pat 2 using the index 12 and retains a record set thus obtained in the record storage part 4. In a step 306, a pointer for a parent node is traced starting with a record having a pointer for a node on the descendant side. That is, the record retrieval part 3 traces a pointer for a parent record of one record in the record set that has satisfied the first condition and has been retained in the record storage part 4 in the previous step 304 and retains the parent record thus obtained in the record storage part 4.

[0059] In the following step 307, it is checked if the designated connection relation is a parent-child relation or not. That is, this step refers to the conditions of connection relation analyzed by the condition analysis part 2 to determine whether it is a parent-child relation or a progenitor-descendant relation. If it is determined as a parent-child relation, the procedure is advanced to a step 310, where it is checked if the retrieval is completed for all the records. If any unprocessed node remains, the procedure is returned to the processing of a record for the next node.

[0060] If it is determined to not be a parent-child relation as a result of the step 307, i.e., the designated connection relation is a progenitor-descendant relation, the procedure is advanced to a step 308 to check if the root node has been traced or not. If the root node has not been traced yet, a pointer for the parent node is traced in a step 309, and the parent node thus obtained is retained in the record storage part 4. Subsequently, the procedure is returned to the step 308 to repeat the procedure to check if the root node has been traced. If it is determined that the root node has been traced, the procedure is advanced to a step 310 to check if the retrieval has been completed for all the records. If any unprocessed node remains, the procedure is returned to the processing of a record for the next node.

[0061] In other words, the node retrieval part 3 determines if a parent record traced by a pointer for a record is a root record (a record having a pointer for a root node) or not. If the parent node thus traced is not a root record, the node retrieval part 3 traces a pointer for the parent record of the above-said parent record and retains the parent record thus obtained in the record storage part 4. This procedure may be repeated. As a result, parent records thus traced one by one will be retained in the record storage part 4.

[0062] If the designated connection relation is a parent-child relation or if a parent record is a root record, it will be checked in the following step 310 whether or not the processing has been completed for all the records in a record set that has satisfied the first condition and has been retained in the record storage part 4. If it is determined that some unprocessed records remain, the procedure is returned to the step 306 to repeat the same processing for the remaining records. If it is confirmed that the processing has been completed for all the records, the procedure is advanced to a step 311 to calculate a product set between a pointer set for the nodes of the records that have satisfied the second condition and have been retained in the record storage part 4 and a pointer set for the nodes of parent (progenitor) records. The result of the calculation of sets satisfies each given condition; therefore, in a step 312, a pointer for a node is traced within the corresponding record, and the node information that satisfies the conditions is read from the data file. Subsequently, in a step 313, the node information thus read is displayed on the display part 7 to complete a series of processes.

[0063] A specific example is explained below in the case that the aforementioned data retrieval processing is performed for data having a tree structure. Fig. 5 is an explanatory view showing an example of data to be retrieved. Here, data to be retrieved having a tree structure is exemplified by document data having a logic tree structure. As shown in Fig. 5, document data to be retrieved is expressed by a hierarchically directed tree. Here, the title of the document (i.e., "report") is the root node that has various types of other nodes such as chapters, clauses and paragraphs. Although only types are shown in Fig. 5, each node has other attributes as well.

[0064] In the case of retrieving data as shown in Fig. 5 using a procedure of the data retrieval processing as shown in Fig. 3 under the condition of "a chapter containing a drawing," the data retrieval processing is performed as shown below. Fig. 6 is a diagram showing the reference relationship of data on a record set in the case of performing the data retrieval processing by following the processing flow as shown in Fig. 3. Fig. 6 should be referred to for the explanation. As shown in Fig. 6, only related portions of data to be retrieved are shown on the right side, and the node 61 (i.e., a block with a hatching) is a node that satisfies the condition. Here, a record set to be referred to is shown on the left side.

[0065] In this data retrieval processing in which the condition is "a chapter containing a drawing," a first condition for nodes is "the type is a chapter," and a second condition for nodes is "the type is a drawing." Also, the connection relation condition is "a progenitor-descendant relation."

[0066] In such a data retrieval processing, the node retrieval part 3 first obtains a group of records (601 ~ 604) having pointers for nodes that satisfy the first condition for nodes, i.e., "the type is a chapter" and a group of records (605 and 606) having pointers for nodes that satisfy the second condition for nodes, i.e., "the type is a drawing."

[0067] Next, the connection relation retrieval part 5 searches for nodes pointed by a pointer 607 for a node relating to a record 605 that satisfies the second condition and a pointer 608 for a node relating to a record 606 that also satisfies the second condition 2 within the subtree index of a group of records (601 ~ 604) that satisfies the first condition.

[0068] In this case, connection relations 62a and pointers for nodes 62b are referred to in the subtree index 62d pointed by subtree retrieval pointers (610 ~ 613) of a group of records (601 ~ 604) to search for nodes commonly pointed by both. As a result, in this case, a node pointed by the pointer 609 in the subtree index 62 pointed by the subtree index pointer 613 agrees with a node pointed by the pointer 608 of the aforementioned record 606. Accordingly, a node that is suitable for the conditions is the node pointed by the pointer 614 for a node relating to the record 604.

[0069] Next, a description of a specific reference to data on records is given below in the case of retrieving data similarly by following the procedure of the data retrieval processing as shown in Fig. 4 under the condition of "a chapter containing a table." As in the previous case, data to be retrieved is also document data having a tree structure as shown in Fig. 5.

[0070] Fig. 7 is a diagram showing the reference relationship of data on each record set in the case of the data retrieval processing by following the processing flow as shown in Fig. 4. Under the condition of "a chapter containing a table," a node is retrieved that agree with the condition. Fig. 7 should be referred to for the explanation. As shown in Fig. 7, only related portions of data to be retrieved are shown on the right side, and nodes 71 and 72 (i.e., blocks with a hatching) are the ones that satisfy the condition. Here, a record set to be referred to is shown on the left side.

[0071] In this data retrieval processing in which the condition is "a chapter containing a table," a first condition is "the type is a chapter," and a second condition is "the type is a table." The connection relation condition is "a progenitor-descendant relation."

[0072] In such a data retrieval processing, the node retrieval part 3 first obtains a group of records (701 ~ 704) having pointers for nodes that satisfy the first condition for nodes, i.e., "the type is a chapter" and a group of records (706 and 707) having pointers for nodes that satisfy the second condition for nodes, i.e., "the type is a table." Next, pointers are traced among a group of those records. Specifically, the pointers for the parent records of the record 706 and the record 707 are traced to obtain a record 701 and a record 705, respectively.

[0073] Here, since the designated connection relation is "a progenitor-descendant relation" and neither the record 701 nor the record 705 thus obtained is not a root record, pointers for parent records are further traced to obtain a record 700 and a record 702. Although the record 700 is a root record, the record 702 is not a root record. Therefore, a pointer for a parent record is further traced from the record 702 to finally obtain a record 700.

[0074] As a result of tracing pointers, a product set is calculated between pointers (709 ~ 712) for nodes corresponding to the records that satisfy the first condition (i.e., "the type is a chapter") and pointers (708, 709, 710 and 713) for nodes within a group of records (700, 701, 702 and 705) obtained by sequentially tracing parent records. As a result of the calculation of the product set, a pointer 709 and a pointer 710 are obtained as pointers for nodes that satisfy the given condition. Accordingly, nodes suitable for the condition are a node 71 and a node 72 pointed by the pointers 709 and 710.

[0075]

[Effects of the Invention] As explained above, in the data retrieval apparatus according to the present invention, at the time of retrieving a node that satisfies a second condition and has a parent-child or progenitor-descendant relation with a node that satisfies a first node from a data set expressed by a tree structure, a subtree index used for the retrieval does not need to contain information on the attribute values of nodes and, therefore, is smaller in size than a transposed file, which leads to a decrease in the memory capacity of the index. Also, retrieving a node that satisfied a second condition and is a parent or a progenitor of a node that satisfies a first condition can be performed only by scanning a record set, resulting in a decrease in the number of steps of reading data into the main storage from a secondary storage. As a result, the throughput of the data retrieval processing can be enhanced as a whole.

[Brief Explanation of the Drawings]

[Fig. 1] Fig. 1 is a block diagram showing the constitution of the main part of the data retrieval apparatus according to one embodiment of the present invention.

[Fig. 2] Fig. 2 is a diagram showing the data structures of a node retrieval file, a subtree retrieval file and a data file and their relationship.

[Fig. 3] Fig. 3 is a flowchart showing an example of data retrieval processing in which a node is retrieved that satisfies a second condition and is a child or a descendant of a node that satisfies a first condition.

[Fig. 4] Fig. 4 is a flowchart showing a procedure for data retrieval processing using a pointer 15 for a parent record of each record in which a node is retrieved that satisfies a second condition and is a parent or a progenitor of a node that satisfies a first condition.

[Fig. 5] Fig. 5 is an explanatory view showing an example of data to be retrieved.

[Fig. 6] Fig. 6 is a diagram showing the reference relation of record set data at a time when data retrieval processing is performed in accordance with the processing flow as shown in Fig. 3.

[Fig. 7] Fig. 7 is a diagram showing the reference relation of record set data at a time when data retrieval processing is performed in accordance with the processing flow as shown in Fig. 4.

[Fig. 8] Fig. 8 is a diagram showing an example of the logic structure of a document using a tree structure.

[Fig. 9] Fig. 9 is a diagram showing a way of using a transposed file for a subtree at the time of the retrieval processing using the parent-child or progenitor-descendant relationship of a node.

[Fig. 10] Fig. 10 is a diagram showing an example of data structure having a path necessary for searching one node for its progenitor node.

[Fig. 11] Fig. 11 is a diagram showing a first mode for continuously allocating node information to secondary memory.

[Fig. 12] Fig. 12 is a diagram showing a second mode for splitting node information to allocate it to secondary information.

[Explanation of Reference Numerals]

1. input part; 2. condition analysis part; 3. node retrieval part; 4. record storage part; 5. connection relation retrieval part; 6. display processing part; 7. display part; 8. node index file; 9. subtree index file; 10. data file; 11. subtree index; 12. index; 13. record; 14. record set; 15. pointers for parent records; 16. pointers for a subtree index; 17. pointers for nodes; 18. connection relation; 19. pointers for nodes; 21. node data;

61. node to be retrieved; 62. subtree index; 62a. connection relation; 62b. pointers for nodes; 71, 72. nodes to be retrieved; 90. root node; 91. A node; 92. B node; 90a. transposed file of the root node; 91a. transposed file of the A node; 92a. transposed file of the B node; 110. node information; 111. pointer for a parent node; 112. pointers for child nodes; 113. node attributes; 121. link information; 121a. pointer for a parent node; 121b. pointer for a child set; 121c. pointer for an attribute set; 122. child set; 123. attribute set; 601 ~ 606. records; 607 ~ 609. pointers for nodes; 610 ~ 613. pointers for a subtree index; 614. pointers for nodes; 700~707. records; 708 ~ 715. pointers for nodes

[Fig. 1]

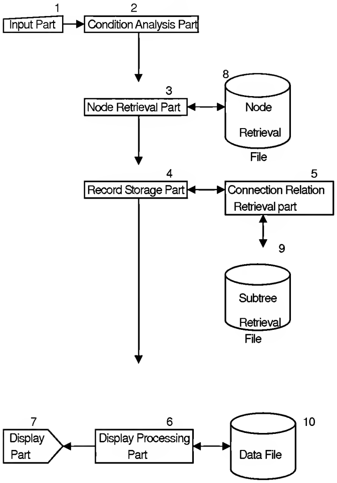


Fig 1. Constitution of a data retrieval apparatus

[Fig. 5]

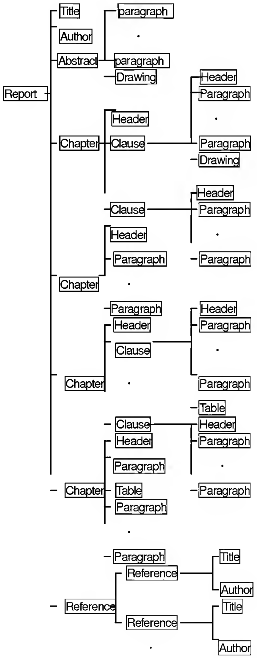
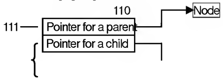


Fig. 5. An Example of data to be retrieved

[Fig. 11]



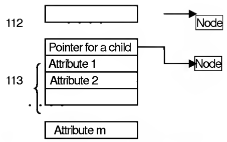


Fig. 11. An example of methods for allocating node information to a storage area

[Fig. 2]

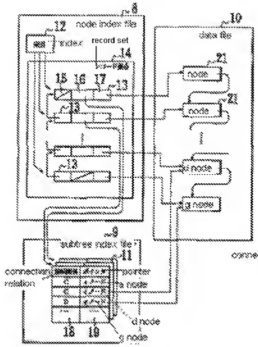


Fig. 2. Data structure of index files

[Fig. 6]

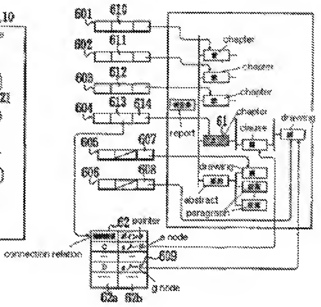


Fig. 6. An example of the data reference relation of a record set

[Fig. 7]

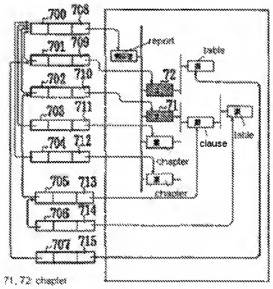


Fig. 7. An example of the data reference relation of a record set

[Fig. 8]

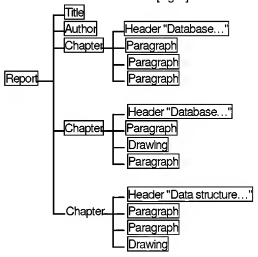


Fig. 8. An example of the logic structure of a document

[Fig. 3]

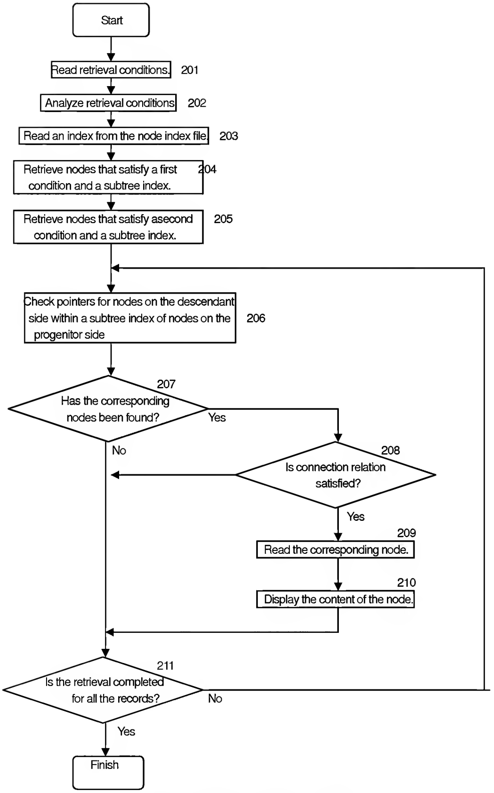
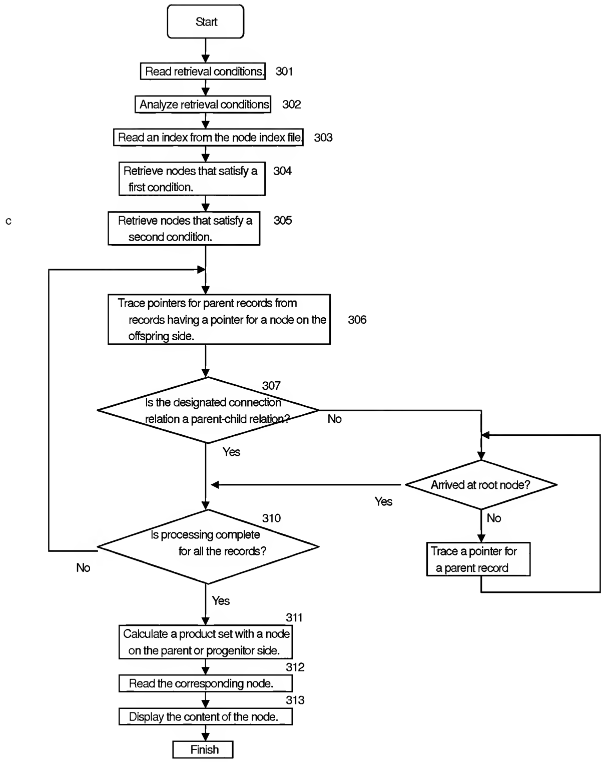

Start
Read retrieval conditions.
201
Analyze retrieval conditions
202
Read an index from the node index file.
203
Retrieve nodes that satisfy a first condition and a subtree index.
204
Retrieve nodes that satisfy asecond condition and a subtree index.
205
Check pointers for nodes on the descendant side within a subtree index of nodes on the progenitor side
206
207
Has the corresponding nodes been found?
Yes
No
208
Is connection relation satisfied?
Yes
209
Read the corresponding node.
210
Display the content of the node.
211
Is the retrieval completed for all the records?
No
Yes
Finish

Fig. 3. A flow chart of data retrieval processing

[Fig. 4]



Start
Read retrieval conditions.
301
Analyze retrieval conditions
302
Read an index from the node index file
303
Retrieve nodes that satisfy a first condition.
304
c
Retrieve nodes that satisfy a second condition.
305
Trace pointers for parent records from records having a pointer for a node on the offspring side.
306
307
Is the designated connection relation a parent-child relation?
No
Yes
Arrived at root node?
Yes
No
Trace a pointer for a parent record
310
Is processing complete for all the records?
No
Yes
311
Calculate a product set with a node on the parent or progenitor side.
312
Read the corresponding node.
313
Display the content of the node.
Finish

Fig. 4. A flowchart of data retrieval processing by searching a record set

[Fig. 9]

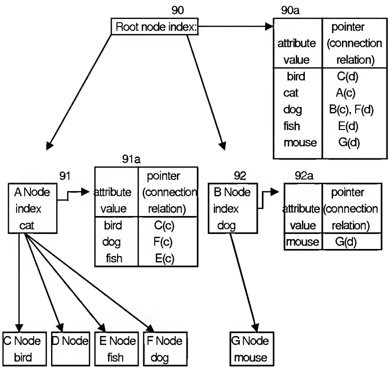


Fig. 9. Retrieval by transposed files

[Fig. 10]

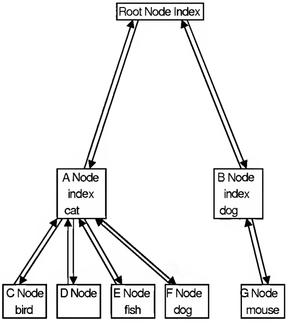


Fig. 10. Retrieval by bilateral links

[Fig. 12]

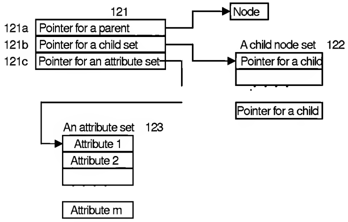


Fig. 12. Another example of methods for allocating node information to a storage area